# WWW16 Zadania kwalifikacyjne
# Tworzenie wyszukiwarki internetowej

Mateusz Sieniawski
1 kwietnia 2020

## Wstęp

1. Jak wysyłać zadania oraz jak się kontaktować z prowadzącym? Najlepiej mailem na adres msieniawski98@gmail.com **koniecznie z tagiem [WWW16] w tytule** oraz podpisując się w mailu. W przypadku drobnych pytań można mnie też łapać na Messengerze.
2. Można i do tego zachęcam wysyłać rozwiązania niepełne. Jeżeli wyślesz swoje rozwiązanie **odpowiednio wcześniej**, to powiem Ci moje uwagi do niego i będziesz miał/a możliwość jego poprawienia.
3. Jako rozwiązanie oczekuję
    a. Kodu źródłowego, oraz
    b. **krótkiego** raportu, w którym opiszesz które z zadań zrobiłeś/aś (albo jeżeli nie uda Ci się zrobić jakiegoś zadania w pełni, to które jego podpunkty zostały zrealizowane) i odpowiedzi na pytania.
4. Punktacja.
    a. W przypadku tych zadań kwalifikacyjnych jest inna niż na innych warsztatach. To, ile punktów zdobędziesz zależy od tego, które zadania oraz jak dobrze robili je inni. Duże punkty, tzn. 10 punktów za każde z trzech dużych zadań, **dzielą się pomiędzy wszystkie osoby, które je zrobią**. Jeżeli dużo osób zrobi jakieś zadanie, to będzie za nie mało punktów. Dlatego optymalna strategia jest taka, aby każdy wybrał w miarę możliwości różne zadania.
    b. Przewiduję, że mogą być oddawane rozwiązania niepełne. Dlatego Twoja liczba punktów za duże zadanie X będzie wynosić (10 pkt) * ai / (a1 + a2 + … + an), gdzie aj - ocena osoby j (liczba od 0 do 1), ai -Twoja ocena za zadanie.
    c. Istnieją premie. Premia to dodatkowe punkty do zdobycia niezależne od punktów za duże zadania, które nie dzielą się pomiędzy innych uczestników. Będą one oznaczone kolorem zielonym.
    d. Twoja ostateczna suma punktów to suma punktów za każde z dużych zadań oraz premii. Np. jeżeli tylko osoby A oraz B zrobią duże zadanie z Pythona, zrobią je dobrze, nie zrobią żadnych innych zadań, a ponadto osoba A uzyskała łącznie 3 punkty premii, to ostateczny wynik osoby A wynosi 8 punktów, a osoby B - 5 punktów.
    e. Aby wspomóc proces wyboru różnych zadań istnieje arkusz, w którym każdy może zaznaczyć, które zadanie chce oddać https://docs.google.com/spreadsheets/d/174OzeLlQPa5BU-id8OYQ3eWCBOsCgz-9Glsh8MuMOXs/edit?usp=sharing

    f. Duże zadanie układają się w logiczną całość (są mniejszymi fragmentami większego systemu), dlatego zachęcam do zrobienia ich wszystkich.
    g. Zdobycie 10 punktów daje gwarancję zakwalifikowania. Oczywiście ostateczny próg może (i prawdopodobnie będzie) niższy.
5. Algorytm postępowania w przypadku napotkania na błąd, z którym nie potrafisz sobie poradzić:
    a. Opisz swój problem tutaj https://stackoverflow.com/. Jeżeli to nie doprowadziło do rozwiązania, to
    b. Opisz swój problem tutaj https://google.com/. Jeżeli to nie doprowadziło do rozwiązania, to
    c. Kup żółtą kaczuszkę do kąpieli. Nadaj jej imię. Opowiedz jej linia po linii, w jaki sposób działa Twój kod. (https://en.wikipedia.org/wiki/Rubber_duck_debugging). Jeżeli to nie doprowadziło do rozwiązania, to
    d. Napisz do mnie.
6. Te zadania są **trudne** i wymagają dużego nakładu czasu. Za to ich zrobienie nauczy Cię wielu rzeczy, w szczególności technologii, które są obecnie powszechnie stosowane. Powodzenia!

# Rozgrzewka

- Opisz **krótko** swoje doświadczenie programistyczne. Jakie technologie znasz? (Za to pytanie nie ma punktów, ale jest dla mnie istotne)
- Wyślij swoje top 10 memów z koronawirusem. (+1 pkt **za najlepsze**)
- Lektury do poczytania. Wystarczy że napiszesz, że przeczytałeś/aś. "*Podstawą cywilizacji jest lenistwo i **zaufanie**.*" ~Maria Mach (+1 pkt)
    - https://www.quora.com/How-do-you-build-a-search-engine-from-scratch-What%E2%80%99s-the-best-technology-stack-for-this
    - https://softwareengineering.stackexchange.com/questions/47360/if-i-wanted-to-build-a-search-engine-how-would-i-start
    - https://azati.ai/build-an-intelligent-search-engine-from-scratch/
    - https://moz.com/beginners-guide-to-seo/how-search-engines-operate
- Internet jest gigantyczny i oczywiście nie będziemy wstanie stworzyć wyszukiwarki przeszukującą cały internet pod kątem każdej możliwej treści. Jaką część internetu chcesz przeszukiwać oraz jakiego typu rzeczy chciałbyś wyszukiwać? Spróbuj wymyślić coś, z czego rzeczywiście byś korzystał/a albo pewne grono osób by korzystało. (+3 pkt **za najlepszą**)

# Duże zadania

## Python (10 pkt)

Napisz crawler maili.
A dokładniej, napisz program w Pythonie, który

1. Wchodzi na losową stronę internetową. (wybranie **losowej** strony internetowej jest trudne, opisz w jaki sposób to osiągnąłeś/aś).
2. Ściąga jej kod źródłowy, a następnie wyciąga wszystkie znajdujące się w nim adresy mailowe i zapisuje do pliku tekstowego.
3. Następnie, wyciąga ze strony wszystkie wszystkie linki. Jeżeli dotychczas zebrano łącznie mniej niż 100 adresów mailowych, to rekurencyjnie przechodzi do każdej ze znalezionych stron i wyciąga z nich adresy mailowe (wróć do punktu 2.)
4. Jeżeli zebrano mniej niż 100 adresów mailowych, a na stronie nie ma żadnych linków, to wylosuj nową stronę (wróć do punktu 1.)

Przetestuj działanie swojego programu. Skomentuj jego działanie w raporcie. Jak długo zajmuje to crawlowanie? Jak często dochodzisz do momentu, gdy na stronie nie ma żadnych linków? Jak można by polepszyć działanie crawlera? Opisz napotkane trudności.

(+1 pkt za napisanie programu współbieżne tzn. można crawlować wiele stron na raz, w osobnych wątkach)
(+1 pkt za przechowywanie kodu źródłowego każdej zcrawlowanej strony w NoSQL-owej bazie danych)

Przydatne materiały:

- https://www.freecodecamp.org/news/how-to-scrape-websites-with-python-and-beautifulsoup-5946935d93fe/
- https://towardsdatascience.com/how-to-web-scrape-with-python-in-4-minutes-bc49186a8460
- https://medium.com/velotio-perspectives/an-introduction-to-asynchronous-programming-in-python-af0189a88bbb

# Flask + Docker (10 pkt)

Odpowiedz na pytania:

- Co to Docker? Po co go stosować?
- Co to Nginx? Po co go stosować?

Stwórz plik konfiguracyjny .yaml dla Docker Compose, po uruchomieniu którego

- Działa aplikacja Flaskowa w kontenerze. Ta aplikacja wyświetla wszystkie pliki znajdujące się w danym katalogu. Do wyglądu należy użyć biblioteki Bootstrap.
- Należy stworzyć kontener Redisa, w którym przechowywana jest informacja, ile razy każdy plik został ściągnięty. Ta informacja ma być wyświetlana w aplikacji flaskowej przy każdym z plików.
- Działa Nginx w kontenerze na porcie 80. Żądania będę przychodzić do nginxa, a następnie będą przekierowywane do Flaska.
- Połączenie będzie po https-ie (można użyć np. Let's encrypt).
- Pliki statyczne będą serwowane bezpośrednio przez Nginx-a.

(+1 pkt za odpalenie Dockera w trybie rozproszonym (na więcej niż jednym hoście) tzn. Docker Swarm. Wystarczą screeny, że kontenery zostały zreplikowane na różne hosty.)
(+1 punkt za uruchomienie tego na AWS - Amazon Web Services. Wystarczy podesłać screeny działającej instancji EC2 oraz działającej aplikacji pod amazonowym adresem)

Przydatne materiały:

- https://docs.docker.com/install/
- https://docs.docker.com/get-started/
- https://docs.docker.com/compose/gettingstarted/
- https://www.w3schools.com/html/
- https://www.w3schools.com/css/
- https://www.w3schools.com/bootstrap4/
- https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world

# Kafka (10 pkt)

Odpowiedz na pytania:

- Co to kafka?
- Do czego się ją stosuje?
- Czym jest komunikacja synchroniczna, a czym komunikacja asynchroniczna? Podaj przykłady zastosowania każdej z nich.

Uruchom Kafkę.

- Jeżeli zrobiłeś/aś duże zadanie z Pythona, to rozdziel go na dwa programy: jeden ściągający kod strony internetowej, a drugi analizujący go tzn. wyciągający adresy mailowe i linki. Komunikacja pomiędzy tymi programami ma się odbywać poprzez Kafkę.
- Jeżeli nie robiłeś/aś dużego zadania z Pythona, to napisz dwa proste programy w Pythonie. Pierwszy z nich w nieskończonej pętli co jakiś czas ściąga kod źródłowy jakiejś strony internetowej, dołącza do niej informacje o obecnym czasie, a następnie przesyła to poprzez Kafkę do drugiego. Drugi program w nieskończonej pętli wyciąga te dane z Kafki i wypisuje na standardowe wyjście.

(+1 pkt za uruchomienie Kafki w trybie rozproszonym tzn. jako klaster na więcej niż jednym hoście)

Przydatne materiały:

- https://www.sohamkamani.com/blog/2017/11/22/how-to-install-and-run-kafka/
- https://towardsdatascience.com/getting-started-with-apache-kafka-in-python-604b3250aa05
- https://github.com/spotify/docker-kafka